Week 4 Lab - Sentiment Analysis II

Clarissa Boyajian

2022-04-25

Question 1: Load and Clean Data

The code below loads the tweet data and sentiment lexicons

```
raw_tweets <- read.csv("https://raw.githubusercontent.com/MaRo406/EDS_231-text-sentiment/main/dat/IPCC_
                       header = TRUE)

# load sentiment lexicons
bing_sent <- get_sentiments('bing')
nrc_sent <- get_sentiments('nrc')
```

The code below cleans the data, including removing all URLs and @ mentions for the tweets.

```
clean_tweets <-
  raw_tweets[, c(4, 6, 10:11)]

tweets <-
  tibble(id = seq(1:length(clean_tweets$Title)),
         text = clean_tweets$Title,
         date = as.Date(clean_tweets$Date, '%m/%d/%y'),
         sentiment = clean_tweets$Sentiment,
         emotion = clean_tweets$Emotion) %>%
  mutate(text = str_replace(string = text,
                            pattern = "http.*[:space:]",
                            replacement = ""),
         text = str_replace(string = text,
                            pattern = "http.*$",
                            replacement = ""),
         text = str_replace(string = text,
                            pattern = "@.*[:space:]",
                            replacement = ""),
         text = str_replace(string = text,
                            pattern = "@.*$",
                            replacement = ""),
         text = str_to_lower(text))
```

The code below splits the tweets out into a dataframe with each word in different row, removes all stop words, adds in the `bing` sentiment and a numberical sentiment score for each word.

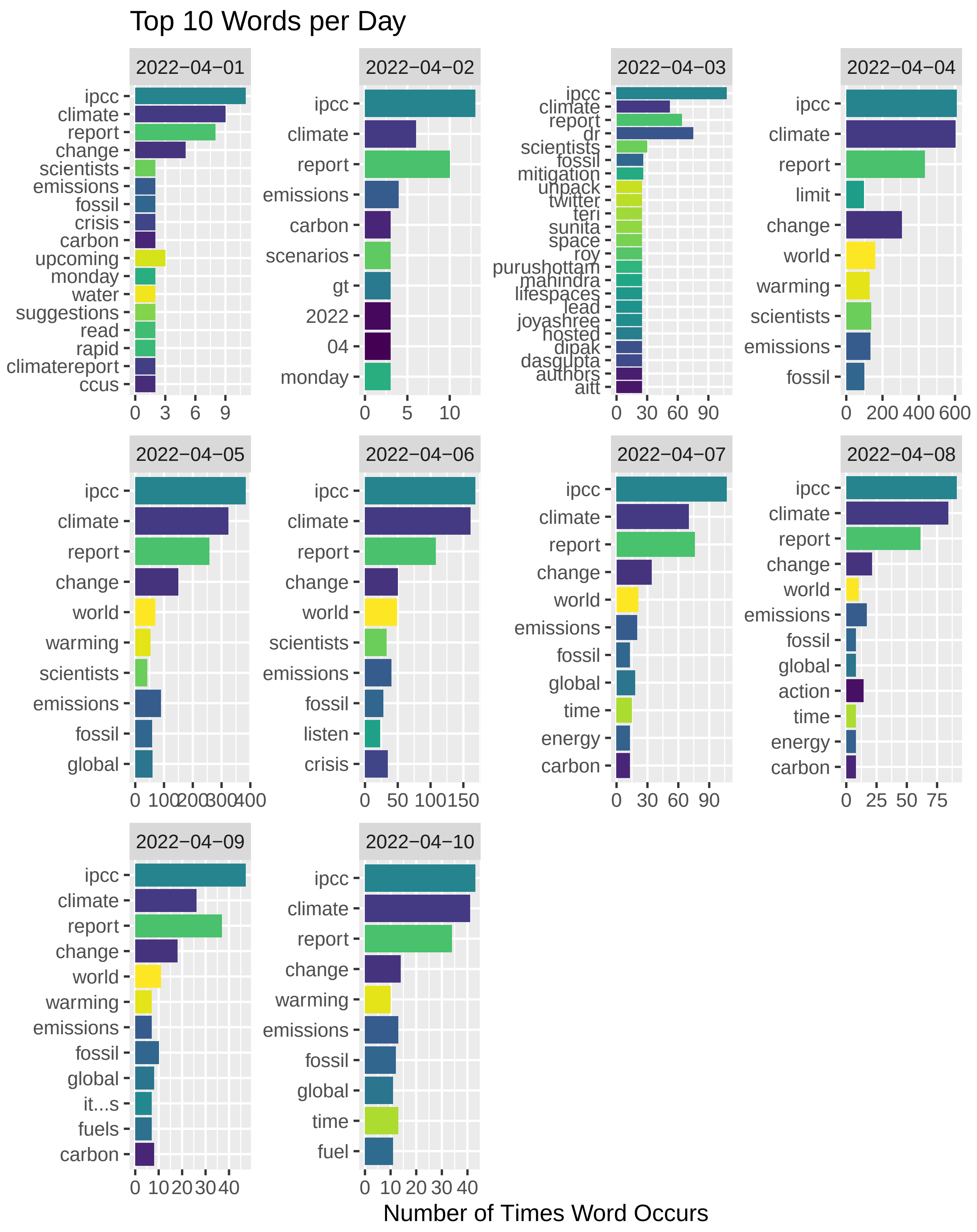
```r
words <- tweets %>%
  select(id, date, text) %>%
  unnest_tokens(output = word,
                input = text,
                token = "words") %>%
  anti_join(stop_words, by = "word") %>%
  left_join(bing_sent, by = "word") %>%
  left_join(
    tribble(
      ~sentiment, ~sent_score,
      "positive", 1,
      "negative", -1),
    by = "sentiment")
```

Question 2: Compare 10 Most Common Words per Day

The code below creates a plot that shows the top 10 words used each day leading up and immediately after to the IPPC report release.

```r
words_common <- words %>%
  group_by(date, word) %>%
  summarise(count = n()) %>%
  group_by(date) %>%
  slice_max(count, n = 10)

ggplot(data = words_common,
       aes(x = count,
           y = reorder(word, count))) +
  geom_col(aes(fill = word)) +
  facet_wrap(~date, scales = "free") +
  guides(fill = "none") +
  scale_fill_viridis_d() +
  labs(x = "Number of Times Word Occurs",
       y = "",
       title = "Top 10 Words per Day")
```

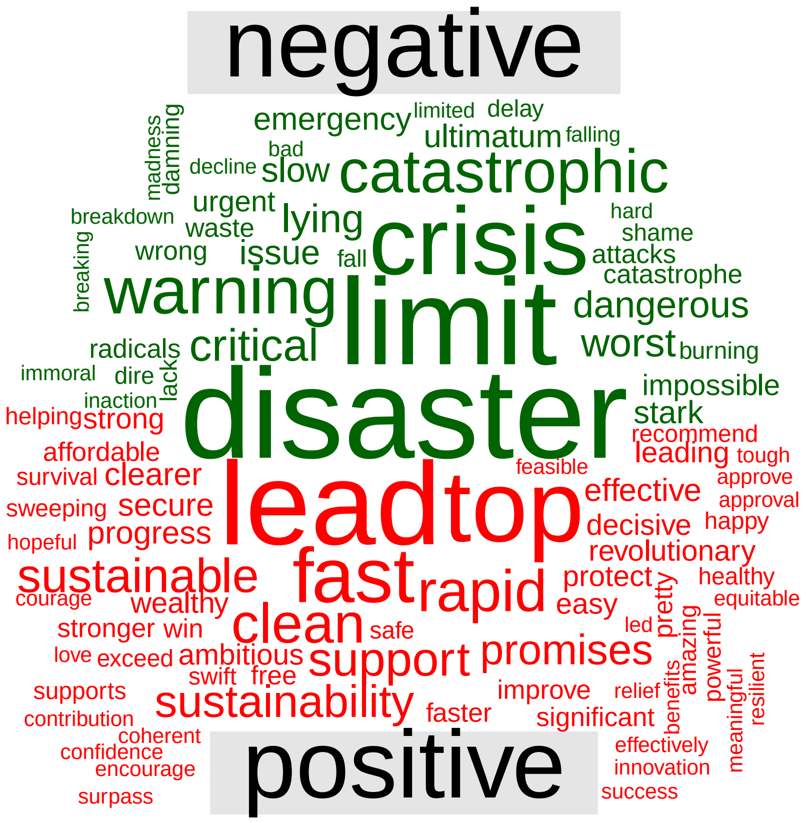


The plot above shows that the top few words each day remain the same across the time period. Interestingly, the total number of mentions per words in the days leading up to release are much lower (around 10 mentions), than in the days immediately after the release (in the 100s), and the days following the release (10-40 mentions).

Question 3: Wordcloud Colored by Sentiment

The code below joins the bing sentiment to the dataframe of words, counts the number of occurrences in each word, and then creates a word cloud where words are colored based on their sentiment.

```
words %>%
  inner_join(get_sentiments("bing")) %>%
  count(word, sentiment, sort = TRUE) %>%
  acast(word ~ sentiment, value.var = "n", fill = 0) %>%
  comparison.cloud(colors = c("darkgreen", "red", "lightgrey"),
                   max.words = 100)
```



Question 4: Most Tagged Accounts

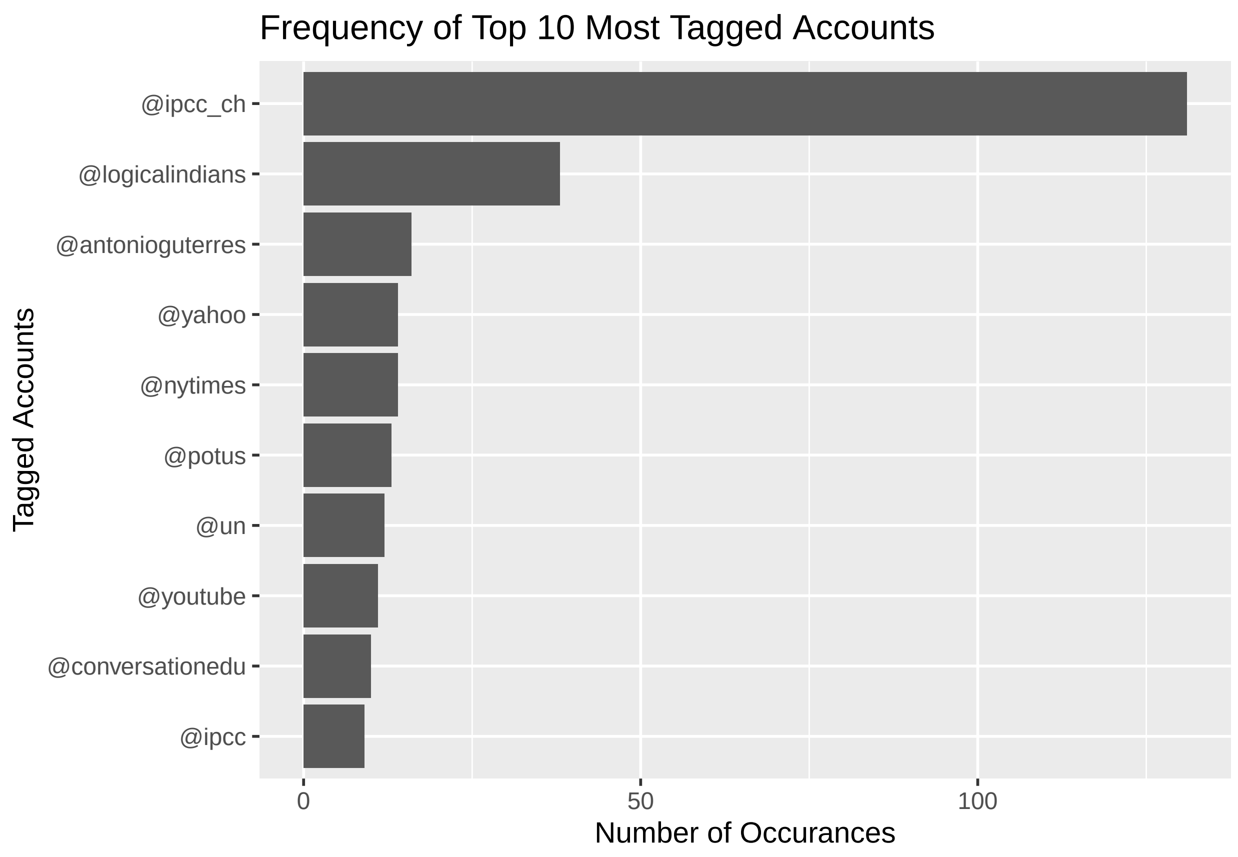
The code below uses the quanteda famliy of packages to clean and wrangle the tweet data. This includes pulling out just the @ mentions within each tweet, tidying the dataframe, and then creating a plot showing the frequency of use for the top 10 @ mentions.

```
corpus <- corpus(clean_tweets$Title)

tagged_tweets <- tokens(corpus, remove_punct = TRUE) %>%
  tokens_keep(pattern = "@*") %>%
  dfm() %>%
  textstat_frequency(n = 10)
```

```r
tagged_tweets_tidy <- tokens(corpus, remove_punct = TRUE) %>%
  tokens_keep(pattern = "@*") %>%
  dfm() %>%
  tidy()

ggplot(data = tagged_tweets,
       aes(x = frequency,
           y = reorder(feature, frequency))) +
  geom_col() +
  labs(x = "Number of Occurances",
       y = "Tagged Accounts",
       title = "Frequency of Top 10 Most Tagged Accounts")
```



Question 5: Sentiment Comparison

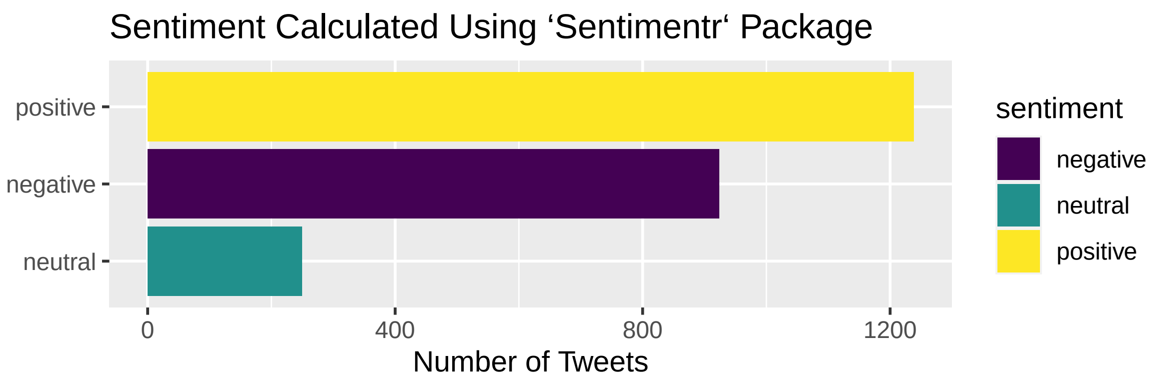
The code below creates two dataframes, one that uses the `sentimentr` package to create a sentiment score, and one that uses the Bandwatch calculated sentiment scores.

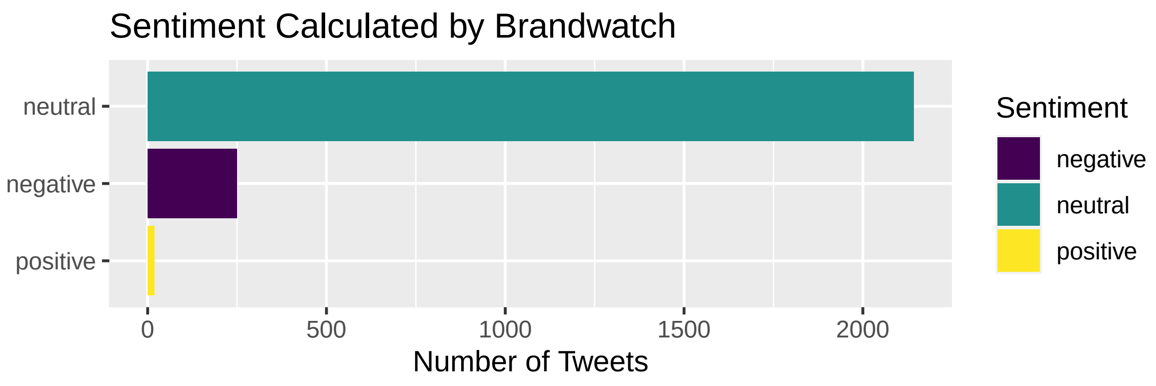
```r
# Calculate sentiment and plot
p <- get_sentences(clean_tweets$Title) %>%
  sentiment() %>%
  group_by(element_id) %>%
  summarize(sentiment_score = mean(sentiment)) %>%
  mutate(sentiment = case_when(sentiment_score < 0 ~ "negative",
                                sentiment_score == 0 ~ "neutral",
                                sentiment_score > 0 ~ "positive")) %>%
  group_by(sentiment) %>%
  summarize(count = n())%>%
  ggplot(aes(x = count,
             y = reorder(sentiment, count),
             fill = sentiment)) +
  geom_col() +
  scale_fill_viridis_d() +
  labs(x = "Number of Tweets",
       y = "",
       title = "Sentiment Calculated Using `Sentimentr` Package")

# Plot Brandwatch sentiment scores
p2 <- clean_tweets %>%
  filter(Sentiment %in% c("positive", "negative", "neutral")) %>%
  group_by(Sentiment) %>%
  summarize(count = n()) %>%
  ggplot(aes(x = count,
             y = reorder(Sentiment, count),
             fill = Sentiment)) +
  geom_col() +
  scale_fill_viridis_d() +
  labs(x = "Number of Tweets",
       y = "",
       title = "Sentiment Calculated by Brandwatch")

(p / p2) +
  plot_annotation(title = "Sentiment of Tweets")
```

Sentiment of Tweets


Sentiment Calculated Using ‘Sentimentr‘ Package
positive
negative
neutral
Number of Tweets
sentiment
negative
neutral
positive


Sentiment Calculated by Brandwatch
neutral
negative
positive
Number of Tweets
Sentiment
negative
neutral
positive